



# UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE  
United States Patent and Trademark Office  
Address: COMMISSIONER FOR PATENTS  
P.O. Box 1450  
Alexandria, Virginia 22313-1450  
[www.uspto.gov](http://www.uspto.gov)

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/513,518	02/25/2000	Cedell Adam Alexander JR.	RAL9-99-0073	7208
45211	7590	07/02/2007		
Robert A. Voigt, Jr. WINSTEAD SECHREST & MINICK PC PO BOX 50784 DALLAS, TX 75201			EXAMINER MEW, KEVIN D	
			ART UNIT 2616	PAPER NUMBER
			MAIL DATE 07/02/2007	DELIVERY MODE PAPER

**Please find below and/or attached an Office communication concerning this application or proceeding.**

The time period for reply, if any, is set in the attached communication.

52

<b>Office Action Summary</b>	<b>Application No.</b> 09/513,518	<b>Applicant(s)</b> ALEXANDER ET AL.	
	<b>Examiner</b> Kevin Mew	<b>Art Unit</b> 2616	

**-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --**

**Period for Reply**

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

**Status**

- 1) ☒ Responsive to communication(s) filed on 20 April 2007.
- 2a) ☐ This action is **FINAL**.                      2b) ☒ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

**Disposition of Claims**

- 4) ☒ Claim(s) 37-39, 41, 42, 45-47, 50, 53-55, 58-60, 63-65, 67 and 68 is/are pending in the application.
- 4a) Of the above claim(s) \_\_\_\_\_ is/are withdrawn from consideration.
- 5) ☒ Claim(s) 39, 42, 47, 50, 53-55, 58-60, 65 and 68 is/are allowed.
- 6) ☒ Claim(s) 37, 38, 41, 45, 46, 63, 64 and 67 is/are rejected.
- 7) ☐ Claim(s) \_\_\_\_\_ is/are objected to.
- 8) ☐ Claim(s) \_\_\_\_\_ are subject to restriction and/or election requirement.

**Application Papers**

- 9) ☐ The specification is objected to by the Examiner.
- 10) ☐ The drawing(s) filed on \_\_\_\_\_ is/are: a) ☐ accepted or b) ☐ objected to by the Examiner.  
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).  
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

**Priority under 35 U.S.C. § 119**

- 12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) ☐ All    b) ☐ Some \* c) ☐ None of:
1. ☐ Certified copies of the priority documents have been received.
  2. ☐ Certified copies of the priority documents have been received in Application No. \_\_\_\_\_.
  3. ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

\* See the attached detailed Office action for a list of the certified copies not received.

**Attachment(s)**

- |  |   |
|--|---|
| 1) <input checked="" type="checkbox"/> Notice of References Cited (PTO-892)                                | 4) <input type="checkbox"/> Interview Summary (PTO-413)<br>Paper No(s)/Mail Date. _____ |
| 2) <input type="checkbox"/> Notice of Draftsperson's Patent Drawing Review (PTO-948)                       | 5) <input type="checkbox"/> Notice of Informal Patent Application                       |
| 3) <input type="checkbox"/> Information Disclosure Statement(s) (PTO/SB/08)<br>Paper No(s)/Mail Date _____ | 6) <input type="checkbox"/> Other: _____  |

***Detailed Action***

***Response to Amendment***

1. Applicant's Remarks/Arguments filed on 4/20/2007 have been considered. Claims 35-36, 40, 43-44, 48-49, 51-52, 56-57, 61-62 and 66 have been cancelled by applicant. Claims 37-39, 41-42, 45-47, 50, 53-55, 58-60, 63-65 and 67-68 are currently pending.

***Claim Rejections - 35 USC § 103***

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

2. Claims 37-38, 41, 45-46, 63-64, 67 are rejected under 35 U.S.C. 103(a) as being unpatentable over Kumar et al. (USP 5,970,069) in view of Gulick et al. (USP 6,314,501).

Regarding claim 37, Kumar discloses a network switch (a network switch, a combination of elements 34, 36, 40, 42, 44, 180, 166, 168, 182, 176, 170, 160, 162, 164, 150, 174, Fig. 5) comprising:

- a CPU (CPU, CW4011 MiniRISC, element 90, Fig. 3);
- a memory system (cache, elements 92, 94, Fig. 3) having circuitry operable to attach to the CPU (having circuitry to attach to CPU 90, Fig. 3);
- a switch fabric system (SWAN, E110 controller, V.34 I/F, 4X Serial, elements 72, 74, 76a, 76b, 76c, 76d, 80, Fig. 3) having circuitry operable to attach to the CPU (having circuitry to attach to CPU 90, Fig. 3);

a port controller (V.34 DAA, element 182, Fig. 5) having circuitry operable to attach to the switch fabric system (to attach to the switch fabric system via line 44, Figs. 3 and 5);

a software application operable to execute on the CPU (CPU executes V.34 modem algorithm, col. 7, lines 47-53);

a Forwarding Database Distribution Library (FDDL) system (memory 84, DRAM 162, Flash ROM 164, comprising a linked list of buffer memory descriptors BMD for each channel, element 84, col. 7, lines 25-40 and Fig. 3) operable to execute on the CPU (to execute on CPU 90, col. 7, lines 25-40, Fig. 3); and

a switch device driver (Multi Channel DMA Controller, element 82, Fig. 3) operable to execute on the CPU (communicates with CPU 90, col. 7, lines 25-40 and Fig. 3),

wherein the software application is operable to communicate with the FDDL system (V.34 communicates with the memory 82 once the channel is identified as V.34 modem channel to receive and perform digital signal processing for the packet, col. 7, lines 47-53), the FDDL system (memory 84, Fig. 3) is operable to communicate with the switch device driver (communicates with Multi Channel DMA Controller 82, Fig. 3), and the switch device driver (Multi Channel DMA Controller 82, Fig. 3) is operable to communicate with the switch fabric (communicates with SWAN, E110 controller, V.34 I/F, 4X Serial, elements 72, 74, 76a, 76b, 76c, 76d, 80, Fig. 3).

Kumar does not explicitly show the FDDL system defines an FDDL API for communication with the software application, and the FDDL system defines a Switch Services API for communication with the switch device driver.

However, Gulick discloses a computer system wherein an API communicates with application software (col. 54, lines 36-42 and Fig. 22) and another API communicates with an Ethernet device driver (switch device driver, col. 56, lines 23-33 and Fig. 24).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to modify the network switch of Kumar with the teaching of Gulick in having an API communicates with application software and another API communicates with an Ethernet device driver such that the FDDL system of Kumar will define an FDDL API for communication with the software application, and the FDDL system defines a Switch Services API for communication with the switch device driver.

The motivation to do so is to use an API to prepare the message for input to a network communications interface when an application program initiates a message send operation using the API and to use another API to allow a TCP/IP module to interface with the switch device driver.

Regarding claim 38, Kumar discloses a network switch, comprising:

a CPU (CPU, CW4011 MiniRISC, element 90, Fig. 3);

a memory system (cache, elements 92, 94, Fig. 3) having circuitry operable to attach to the CPU (having circuitry to attach to CPU 90, Fig. 3);

a switch fabric system (SWAN, E110 controller, V.34 I/F, 4X Serial, elements 72, 74, 76a, 76b, 76c, 76d, 80, Fig. 3) having circuitry operable to attach to the CPU (having circuitry to attach to CPU 90, Fig. 3);

a port controller (V.34 DAA, element 182, Fig. 5) having circuitry operable to attach to the switch fabric system (to attach to the switch fabric system via line 44, Figs. 3 and 5);

a software application operable to execute on the CPU (CPU executes V.34 modem algorithm, col. 7, lines 47-53);

a Forwarding Database Distribution Library (FDDL) system (memory 84, DRAM 162, Flash ROM 164, comprising a linked list of buffer memory descriptors BMD for each channel, element 84, col. 7, lines 25-40 and Fig. 3) operable to execute on the CPU (to execute on CPU 90, col. 7, lines 25-40, Fig. 3); and

a switch device driver (Multi Channel DMA Controller, element 82, Fig. 3) operable to execute on the CPU (communicates with CPU 90, col. 7, lines 25-40 and Fig. 3),

a second software application operable to execute on the CPU (CPU executes routing software to retrieve packets stored in local memory and determine their destination, col. 8, lines 52-57), wherein the second software application communicates with the FDDL system (routing software communicates with local memory, col. 7, lines 52-57 and Fig. 3).

wherein the software application is operable to communicate with the FDDL system (V.34 communicates with the memory 82 once the channel is identified as V.34 modem channel to receive and perform digital signal processing for the packet, col. 7, lines 47-53), the FDDL system (memory 84, Fig. 3) is operable to communicate with the switch device driver (communicates with Multi Channel DMA Controller 82, Fig. 3), and the switch device driver (Multi Channel DMA Controller 82, Fig. 3) is operable to communicate with the switch fabric (communicates with SWAN, E110 controller, V.34 I/F, 4X Serial, elements 72, 74, 76a, 76b, 76c, 76d, 80, Fig. 3).

Kumar does not explicitly show the FDDL system defines an FDDL API for communication with the software application, and the FDDL system defines a Switch Services API for communication with the switch device driver.

However, Gulick discloses a computer system wherein an API communicates with application software (col. 54, lines 36-42 and Fig. 22) and another API communicates with an Ethernet device driver (switch device driver, col. 56, lines 23-33 and Fig. 24).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to modify the network switch of Kumar with the teaching of Gulick in having an API communicates with application software and another API communicates with an Ethernet device driver such that the FDDL system of Kumar will define an FDDL API for communication with the software application, and the FDDL system defines a Switch Services API for communication with the switch device driver.

The motivation to do so is to use an API to prepare the message for input to a network communications interface when an application program initiates a message send operation using the API and to use another API to allow a TCP/IP module to interface with the switch device driver.

Regarding claim 41, Kumar discloses the network switch of claim 35 further comprising:  
a CPU (CPU, CW4011 MiniRISC, element 90, Fig. 3);  
a memory system (cache, elements 92, 94, Fig. 3) having circuitry operable to attach to the CPU (having circuitry to attach to CPU 90, Fig. 3);

Art Unit: 2616

a switch fabric system (SWAN, E110 controller, V.34 I/F, 4X Serial, elements 72, 74, 76a, 76b, 76c, 76d, 80, Fig. 3) having circuitry operable to attach to the CPU (having circuitry to attach to CPU 90, Fig. 3);

a port controller (V.34 DAA, element 182, Fig. 5) having circuitry operable to attach to the switch fabric system (to attach to the switch fabric system via line 44, Figs. 3 and 5);

a software application operable to execute on the CPU (CPU executes V.34 modem algorithm, col. 7, lines 47-53);

a Forwarding Database Distribution Library (FDDL) system (memory 84, DRAM 162, Flash ROM 164, comprising a linked list of buffer memory descriptors BMD for each channel, element 84, col. 7, lines 25-40 and Fig. 3) operable to execute on the CPU (to execute on CPU 90, col. 7, lines 25-40, Fig. 3); and

a switch device driver (Multi Channel DMA Controller, element 82, Fig. 3) operable to execute on the CPU (communicates with CPU 90, col. 7, lines 25-40 and Fig. 3),

an independent software application (routing software) operable to execute on the CPU (executes on the CPU, col. 8, lines 47-57); and

an independent software application shim (V.34 modem algorithm) operable to execute on the CPU (executes on the CPU, col. 7, lines 47-53),

wherein the software application is operable to communicate with the FDDL system (V.34 communicates with the memory 82 once the channel is identified as V.34 modem channel to receive and perform digital signal processing for the packet, col. 7, lines 47-53), the FDDL system (memory 84, Fig. 3) is operable to communicate with the switch device driver (communicates with Multi Channel DMA Controller 82, Fig. 3), and the switch device driver

(Multi Channel DMA Controller 82, Fig. 3) is operable to communicate with the switch fabric (communicates with SWAN, E110 controller, V.34 I/F, 4X Serial, elements 72, 74, 76a, 76b, 76c, 76d, 80, Fig. 3);

wherein an independent software application (routing software) communicates with the independent software application shim (communicates with V.34 modem algorithm, col. 8, lines 47-57) and the independent software application shim (V.34 modem algorithm) communicates with the switch device driver (communicates with DMA controller, col. 7, lines 25-40, 47-53); and

a second software application operable to execute on the CPU (CPU executes routing software to retrieve packets stored in local memory and determine their destination, col. 8, lines 52-57).

Kumar does not explicitly show the FDDL system defines an FDDL API for communication with the software application, and the FDDL system defines a Switch Services API for communication with the switch device driver.

However, Gulick discloses a computer system wherein an API communicates with application software (col. 54, lines 36-42 and Fig. 22) and another API communicates with an Ethernet device driver (switch device driver, col. 56, lines 23-33 and Fig. 24).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to modify the network switch of Kumar with the teaching of Gulick in having an API communicates with application software and another API communicates with an Ethernet device driver such that the FDDL system of Kumar will define an FDDL API for

communication with the software application, and the FDDL system defines a Switch Services API for communication with the switch device driver.

The motivation to do so is to use an API to prepare the message for input to a network communications interface when an application program initiates a message send operation using the API and to use another API to allow a TCP/IP module to interface with the switch device driver.

Regarding claim 45, Kumar discloses a network switch comprising:

a CPU (CPU, CW4011 MiniRISC, element 90, Fig. 3);

a memory system (cache, elements 92, 94, Fig. 3) having circuitry operable to attach to the CPU (having circuitry to attach to CPU 90, Fig. 3);

a switch fabric system (SWAN, E110 controller, V.34 I/F, 4X Serial, elements 72, 74, 76a, 76b, 76c, 76d, 80, Fig. 3) having circuitry operable to attach to the CPU (having circuitry to attach to CPU 90, Fig. 3);

a port controller (V.34 DAA, element 182, Fig. 5) having circuitry operable to attach to the switch fabric system (to attach to the switch fabric system via line 44, Figs. 3 and 5);

a protocol means (V.34 modem algorithm) for providing a service to a network system (for providing data packets being transferred through V.34 interface controller);

a Forwarding Database Distribution Library (FDDL) means (memory 84, DRAM 162, Flash ROM 164, comprising a linked list of buffer memory descriptors BMD for each channel, element 84, col. 7, lines 25-40 and Fig. 3) for communicating with the protocol means (for

identifying V.34 channel data packets with the V.34 modem algorithm, col. 7, lines 25-40, 47-53); and

a switch device driver means (Multi Channel DMA Controller, element 82, Fig. 3) for communicating with the FDDL means (communicating with memory 84, col. 7, lines 25-40) and the port controller (communicates with V.34 CODEC of Fig. 5 via controller 72 of Fig. 3).

Kumar does not explicitly show the FDDL system defines an FDDL API for communication with the software application, and the FDDL system defines a Switch Services API for communication with the switch device driver.

However, Gulick discloses a computer system wherein an API communicates with application software (col. 54, lines 36-42 and Fig. 22) and another API communicates with an Ethernet device driver (switch device driver, col. 56, lines 23-33 and Fig. 24).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to modify the network switch of Kumar with the teaching of Gulick in having an API communicates with application software and another API communicates with an Ethernet device driver such that the FDDL system of Kumar will define an FDDL API for communication with the software application, and the FDDL system defines a Switch Services API for communication with the switch device driver.

The motivation to do so is to use an API to prepare the message for input to a network communications interface when an application program initiates a message send operation using the API and to use another API to allow a TCP/IP module to interface with the switch device driver.

Regarding claim 46, Kumar discloses a network switch comprising:

a CPU (CPU, CW4011 MiniRISC, element 90, Fig. 3);

a memory system (cache, elements 92, 94, Fig. 3) having circuitry operable to attach to the CPU (having circuitry to attach to CPU 90, Fig. 3);

a switch fabric system (SWAN, E110 controller, V.34 I/F, 4X Serial, elements 72, 74, 76a, 76b, 76c, 76d, 80, Fig. 3) having circuitry operable to attach to the CPU (having circuitry to attach to CPU 90, Fig. 3);

a port controller (V.34 DAA, element 182, Fig. 5) having circuitry operable to attach to the switch fabric system (to attach to the switch fabric system via line 44, Figs. 3 and 5);

a protocol means (V.34 modem algorithm) for providing a service to a network system (for providing data packets being transferred through V.34 interface controller);

a Forwarding Database Distribution Library (FDDL) means (memory 84, DRAM 162, Flash ROM 164, comprising a linked list of buffer memory descriptors BMD for each channel, element 84, col. 7, lines 25-40 and Fig. 3) for communicating with the protocol means (for identifying V.34 channel data packets with the V.34 modem algorithm, col. 7, lines 25-40, 47-53); and

a switch device driver means (Multi Channel DMA Controller, element 82, Fig. 3) for communicating with the FDDL means (communicating with memory 84, col. 7, lines 25-40) and the port controller (communicates with V.34 CODEC of Fig. 5 via controller 72 of Fig. 3); and

a second protocol means (serial WAN (SWAN), elements 76a, 76b, 76c, 76d, Fig. 3) for providing a second service to the network system (providing serial WAN service, Fig. 3),

wherein the FDDL means communicates with the second protocol means (memory 84 communicates with serial WAN (SWAN), Fig. 3).

Kumar does not explicitly show the FDDL system defines an FDDL API for communication with the software application, and the FDDL system defines a Switch Services API for communication with the switch device driver.

However, Gulick discloses a computer system wherein an API communicates with application software (col. 54, lines 36-42 and Fig. 22) and another API communicates with an Ethernet device driver (switch device driver, col. 56, lines 23-33 and Fig. 24).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to modify the network switch of Kumar with the teaching of Gulick in having an API communicates with application software and another API communicates with an Ethernet device driver such that the FDDL system of Kumar will define an FDDL API for communication with the software application, and the FDDL system defines a Switch Services API for communication with the switch device driver.

The motivation to do so is to use an API to prepare the message for input to a network communications interface when an application program initiates a message send operation using the API and to use another API to allow a TCP/IP module to interface with the switch device driver.

Regarding claim 63, Kumar discloses a network system comprising:

a network switch (a network switch, a combination of elements 34, 36, 40, 42, 44, 180, 166, 168, 182, 176, 170, 160, 162, 164, 150, 174, Fig. 5) comprising a CPU (CPU, CW4011 MiniRISC, element 90, Fig. 3);

a memory system (cache, elements 92, 94, Fig. 3) having circuitry operable to attach to the CPU (having circuitry to attach to CPU 90, Fig. 3);

a switch fabric system (SWAN, E110 controller, V.34 I/F, 4X Serial, elements 72, 74, 76a, 76b, 76c, 76d, 80, Fig. 3) having circuitry operable to attach to the CPU (having circuitry to attach to CPU 90, Fig. 3);

a port controller (V.34 DAA, element 182, Fig. 5) having circuitry operable to attach to the switch fabric system (to attach to the switch fabric system via line 44, Figs. 3 and 5);

a software application operable to execute on the CPU (CPU executes V.34 modem algorithm, col. 7, lines 47-53);

a Forwarding Database Distribution Library (FDDL) system (memory 84, DRAM 162, Flash ROM 164, comprising a linked list of buffer memory descriptors BMD for each channel, element 84, col. 7, lines 25-40 and Fig. 3, comprising a linked list of buffer memory descriptors BMD for each channel, element 84, col. 7, lines 25-40 and Fig. 3) operable to execute on the CPU (to execute on CPU 90, col. 7, lines 25-40, Fig. 3); and

a switch device driver (Multi Channel DMA Controller, element 82, Fig. 3) operable to execute on the CPU (communicates with CPU 90, col. 7, lines 25-40 and Fig. 3), wherein the software application is operable to communicate with the FDDL system (V.34 communicates with the memory 82 once the channel is identified as V.34 modem channel to receive and

Art Unit: 2616

perform digital signal processing for the packet, col. 7, lines 47-53), the FDDL system (memory 84, Fig. 3) is operable to communicate with the switch device driver (communicates with Multi Channel DMA Controller 82, Fig. 3), and the switch device driver (Multi Channel DMA Controller 82, Fig. 3) is operable to communicate with the switch fabric (communicates with SWAN, E110 controller, V.34 I/F, 4X Serial, elements 72, 74, 76a, 76b, 76c, 76d, 80, Fig. 3).

a backbone (Internet, element 54, Fig. 2b); and

a workstation (workstation, element 52a, Fig. 2b),

wherein the workstation is logically connected to the backbone (workstation is logically connected to the Internet, Fig. 2b), and

wherein the backbone is logically connected to the port controller of the network switch (Internet is logically connected to the port controller V.34 DAA, Fig. 5 of the network switch).

Kumar does not explicitly show the FDDL system defines an FDDL API for communication with the software application, and the FDDL system defines a Switch Services API for communication with the switch device driver.

However, Gulick discloses a computer system wherein an API communicates with application software (col. 54, lines 36-42 and Fig. 22) and another API communicates with an Ethernet device driver (switch device driver, col. 56, lines 23-33 and Fig. 24).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to modify the network switch of Kumar with the teaching of Gulick in having an API communicates with application software and another API communicates with an Ethernet device driver such that the FDDL system of Kumar will define an FDDL API for

communication with the software application, and the FDDL system defines a Switch Services API for communication with the switch device driver.

The motivation to do so is to use an API to prepare the message for input to a network communications interface when an application program initiates a message send operation using the API and to use another API to allow a TCP/IP module to interface with the switch device driver.

Regarding claim 64, Kumar discloses a network system comprising:

a network switch (a network switch, a combination of elements 34, 36, 40, 42, 44, 180, 166, 168, 182, 176, 170, 160, 162, 164, 150, 174, Fig. 5) comprising a CPU (CPU, CW4011 MiniRISC, element 90, Fig. 3);

a memory system (cache, elements 92, 94, Fig. 3) having circuitry operable to attach to the CPU (having circuitry to attach to CPU 90, Fig. 3);

a switch fabric system (SWAN, E110 controller, V.34 I/F, 4X Serial, elements 72, 74, 76a, 76b, 76c, 76d, 80, Fig. 3) having circuitry operable to attach to the CPU (having circuitry to attach to CPU 90, Fig. 3);

a port controller (V.34 DAA, element 182, Fig. 5) having circuitry operable to attach to the switch fabric system (to attach to the switch fabric system via line 44, Figs. 3 and 5);

a software application operable to execute on the CPU (CPU executes V.34 modem algorithm, col. 7, lines 47-53);

a Forwarding Database Distribution Library (FDDL) system (memory 84, DRAM 162, Flash ROM 164, comprising a linked list of buffer memory descriptors BMD for each channel,

element 84, col. 7, lines 25-40 and Fig. 3, comprising a linked list of buffer memory descriptors BMD for each channel, element 84, col. 7, lines 25-40 and Fig. 3) operable to execute on the CPU (to execute on CPU 90, col. 7, lines 25-40, Fig. 3); and

a switch device driver (Multi Channel DMA Controller, element 82, Fig. 3) operable to execute on the CPU (communicates with CPU 90, col. 7, lines 25-40 and Fig. 3), wherein the software application is operable to communicate with the FDDL system (V.34 communicates with the memory 82 once the channel is identified as V.34 modem channel to receive and perform digital signal processing for the packet, col. 7, lines 47-53), the FDDL system (memory 84, Fig. 3) is operable to communicate with the switch device driver (communicates with Multi Channel DMA Controller 82, Fig. 3), and the switch device driver (Multi Channel DMA Controller 82, Fig. 3) is operable to communicate with the switch fabric (communicates with SWAN, E110 controller, V.34 I/F, 4X Serial, elements 72, 74, 76a, 76b, 76c, 76d, 80, Fig. 3).

a backbone (Internet, element 54, Fig. 2b); and

a workstation (workstation, element 52a, Fig. 2b),

a second software application operable to execute on the CPU (CPU executes routing software to retrieve packets stored in local memory and determine their destination, col. 8, lines 52-57), wherein the second software application communicates with the FDDL system (routing software communicates with local memory, col. 7, lines 52-57 and Fig. 3).

wherein the workstation is logically connected to the backbone (workstation is logically connected to the Internet, Fig. 2b), and

wherein the backbone is logically connected to the port controller of the network switch (Internet is logically connected to the port controller V.34 DAA, Fig. 5 of the network switch).

Art Unit: 2616

Kumar does not explicitly show the FDDL system defines an FDDL API for communication with the software application, and the FDDL system defines a Switch Services API for communication with the switch device driver.

However, Gulick discloses a computer system wherein an API communicates with application software (col. 54, lines 36-42 and Fig. 22) and another API communicates with an Ethernet device driver (switch device driver, col. 56, lines 23-33 and Fig. 24).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to modify the network switch of Kumar with the teaching of Gulick in having an API communicates with application software and another API communicates with an Ethernet device driver such that the FDDL system of Kumar will define an FDDL API for communication with the software application, and the FDDL system defines a Switch Services API for communication with the switch device driver.

The motivation to do so is to use an API to prepare the message for input to a network communications interface when an application program initiates a message send operation using the API and to use another API to allow a TCP/IP module to interface with the switch device driver.

In claim 67, Kumar discloses the network system of claim 61 further comprising:

a network switch (a network switch, a combination of elements 34, 36, 40, 42, 44, 180, 166, 168, 182, 176, 170, 160, 162, 164, 150, 174, Fig. 5) comprising a CPU (CPU, CW4011 MiniRISC, element 90, Fig. 3);

a memory system (cache, elements 92, 94, Fig. 3) having circuitry operable to attach to the CPU (having circuitry to attach to CPU 90, Fig. 3);

a switch fabric system (SWAN, E110 controller, V.34 I/F, 4X Serial, elements 72, 74, 76a, 76b, 76c, 76d, 80, Fig. 3) having circuitry operable to attach to the CPU (having circuitry to attach to CPU 90, Fig. 3);

a port controller (V.34 DAA, element 182, Fig. 5) having circuitry operable to attach to the switch fabric system (to attach to the switch fabric system via line 44, Figs. 3 and 5);

a software application operable to execute on the CPU (CPU executes V.34 modem algorithm, col. 7, lines 47-53);

a Forwarding Database Distribution Library (FDDL) system (memory 84, DRAM 162, Flash ROM 164, comprising a linked list of buffer memory descriptors BMD for each channel, element 84, col. 7, lines 25-40 and Fig. 3, comprising a linked list of buffer memory descriptors BMD for each channel, element 84, col. 7, lines 25-40 and Fig. 3) operable to execute on the CPU (to execute on CPU 90, col. 7, lines 25-40, Fig. 3); and

a switch device driver (Multi Channel DMA Controller, element 82, Fig. 3) operable to execute on the CPU (communicates with CPU 90, col. 7, lines 25-40 and Fig. 3), wherein the software application is operable to communicate with the FDDL system (V.34 communicates with the memory 82 once the channel is identified as V.34 modem channel to receive and perform digital signal processing for the packet, col. 7, lines 47-53), the FDDL system (memory 84, Fig. 3) is operable to communicate with the switch device driver (communicates with Multi Channel DMA Controller 82, Fig. 3), and the switch device driver (Multi Channel DMA

Controller 82, Fig. 3) is operable to communicate with the switch fabric (communicates with SWAN, E110 controller, V.34 I/F, 4X Serial, elements 72, 74, 76a, 76b, 76c, 76d, 80, Fig. 3).

a backbone (Internet, element 54, Fig. 2b); and

a workstation (workstation, element 52a, Fig. 2b),

an independent software application (routing software) operable to execute on the CPU (executes on the CPU, col. 8, lines 47-57); and

an independent software application shim (V.34 modem algorithm) operable to execute on the CPU (executes on the CPU, col. 7, lines 47-53),

wherein the workstation is logically connected to the backbone (workstation is logically connected to the Internet, Fig. 2b), and

wherein the backbone is logically connected to the port controller of the network switch (Internet is logically connected to the port controller V.34 DAA, Fig. 5 of the network switch).

wherein an independent software application (routing software) communicates with the independent software application shim (communicates with V.34 modem algorithm, col. 8, lines 47-57) and the independent software application shim (V.34 modem algorithm) communicates with the switch device driver (communicates with DMA controller, col. 7, lines 25-40, 47-53).

a second software application operable to execute on the CPU (CPU executes routing software to retrieve packets stored in local memory and determine their destination, col. 8, lines 52-57).

Kumar does not explicitly show the FDDL system defines an FDDL API for communication with the software application, and the FDDL system defines a Switch Services API for communication with the switch device driver.

However, Gulick discloses a computer system wherein an API communicates with application software (col. 54, lines 36-42 and Fig. 22) and another API communicates with an Ethernet device driver (switch device driver, col. 56, lines 23-33 and Fig. 24).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to modify the network switch of Kumar with the teaching of Gulick in having an API communicates with application software and another API communicates with an Ethernet device driver such that the FDDL system of Kumar will define an FDDL API for communication with the software application, and the FDDL system defines a Switch Services API for communication with the switch device driver.

The motivation to do so is to use an API to prepare the message for input to a network communications interface when an application program initiates a message send operation using the API and to use another API to allow a TCP/IP module to interface with the switch device driver.

***Allowable Subject Matter***

3. Claims 39, 42, 47, 50, 53-55, 58-60, 65, 68 are allowed.

The following is a statement of reasons for the indication of allowable subject matter:

In claim 39, the network switch of claim 36 wherein the FDDL system comprises:

a base FDDL system;

a software application tower FDDL system; and

a second software application tower FDDL system wherein the base FDDL system communicates with the switch device driver, the software application communicates with the

Art Unit: 2616

software application tower FDDL system, the second software application communicates with the second software application tower FDDL system, and the base FDDL system communicates with the software application tower FDDL system and the second software application tower FDDL system.

In claim 42, the network switch of claim 40 wherein the FDDL system comprises:

a base FDDL system;

a software application tower FDDL system; and

a second software application tower FDDL system wherein the base FDDL system communicates with the switch device driver, the software application communicates with the software application tower FDDL system, the second software application communicates with the second software application tower FDDL system, and the base FDDL system communicates with the software application tower FDDL system and the second software application tower FDDL system.

In claim 47, the network switch of claim 44 wherein the FDDL means comprises:

a base FDDL means for communicating with the switch device driver means;

a protocol tower FDDL means for communicating with the protocol means and the base FDDL means; and

a second protocol tower FDDL means for communicating with a second protocol means and the base FDDL means.

In claim 50, the network switch of claim 48 wherein the FDDL means comprises:

a base FDDL means for communicating with the switch device driver means;

a protocol tower FDDL means for communicating with the protocol means and the base FDDL means; and

a second protocol tower FDDL means for communicating with the second protocol means and the base FDDL means.

In claim 53, the method of claim 52 wherein all communicating between the switch device driver to the FDDL is done through a switch services API; and

all communicating from the FDDL to the first protocol client and the second protocol client is done through an FDDL API.

In claim 54, the method of claim 52 further comprising the steps of:

defining a switch services API for communication between the switch device driver;

and defining an FDDL API for communication between the first protocol client and the FDDL.

In claim 55, the method of claim 52 further comprising the steps:

receiving the information from the switch device driver at an FDDL base within the FDDL;

passing the information from the FDDL base to a first protocol FDDL tower within the FDDL; and

sending the information from the first protocol FDDL tower to the first protocol client.

Regarding claim 58, the computer-readable medium of claim 57 wherein all communicating between the switch device driver to the FDDL is done through a switch services API; and

all communicating from the FDDL to the first protocol client and the second protocol client is done through an FDDL API.

In claim 59, the computer-readable medium of claim 57 having further stored thereon computer-executable instructions for performing the steps comprising:

defining a switch services API for communication between the switch device driver; and  
defining an FDDL API for communication between the first protocol client and the FDDL.

In claim 60, the computer-readable medium of claim 57 having further stored thereon computer-executable instructions for performing the steps comprising:

receiving the information from the switch device driver at an FDDL base within the FDDL;

passing the information from the FDDL base to a first protocol FDDL tower within the FDDL; and

sending the information from the first protocol FDDL tower to the first protocol client.

In claim 65, the network system of claim 62 wherein the FDDL system comprises:

a base FDDL system;

a software application tower FDDL system; and

a second software application tower FDDL system wherein the base FDDL system communicates with the switch device driver, the software application communicates with the software application tower FDDL system, the second software application communicates with the second software application tower FDDL system, and the base FDDL system communicates with the software application tower FDDL system and the second software application tower FDDL system.

In claim 68, the network system of claim 66 wherein the FDDL system comprises:

a base FDDL system;

a software application tower FDDL system; and

a second software application tower FDDL system wherein the base FDDL system communicates with the switch device driver, the software application communicates with the software application tower FDDL system, the second software application communicates with the second software application tower FDDL system, and the base FDDL system communicates with the software application tower FDDL system and the second software application tower FDDL system.

***Response to Arguments***

4. Applicant's arguments filed on 4/20/2007 with respect to claims 37-38, 41, 45-46, 63-64, 67 have been considered but are moot in view of the new ground(s) of rejection.

***Conclusion***

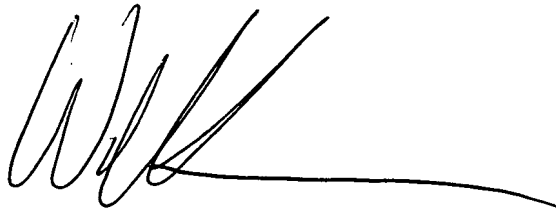
5. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Kevin Mew whose telephone number is 571-272-3141. The examiner can normally be reached on 9:00 am - 5:30 pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Chi Pham can be reached on 571-272-3179. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

Kevin Mew  
Work Group 2616

*Km*

  
WELLINGTON CHIN  
PRIMARY PATENT EXAMINER